PAUL STEPHENS

It's been a stay-at-home month for me – literally at home, ensconced with a Dell 320LX PC, a copy of *dBASE IV 1.1 Developers Edition* and the design for my multi-user version of the PC PLUS magazine planning system.

I already had a database application running, enabling us to keep details of who was writing what for which issues of the magazine, but it was strictly single user – only one person could load the program at a time. We needed a multi-user version, with proper record locking (to stop two people changing a record at the same time) and a few extra features, to run on our new office network.

I'd decided to switch to *dBASE IV* from our trusty old *dBXL* database package as *dB IV* bristles with up-to-date labour-saving development features - and development with the minumum of labour was my prime objective.

Initially, I planned to use *dBASE IV's* Application Generator to produce a simple system with the minimum hands-on programming. The Generator is a tool for designing linked sets of bar and pop-up menus, with 'hooks' for you to call your own screen formats, reports and programs; satisfied that I could get the results I wanted, I was set to go ahead.

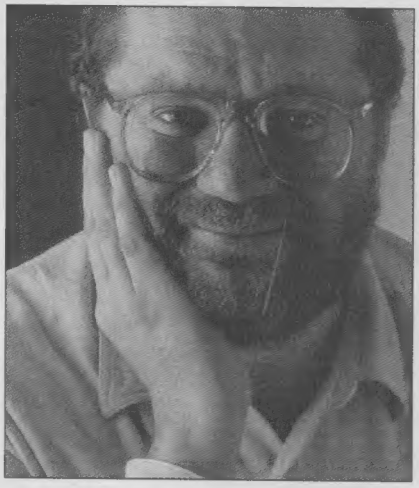
Luckily, I tried generating a sample program first – it churned out nearly 1,000 lines of *dBASE* program code, that ran very slowly on the stand-alone Dell and was positively immobile across the network. So, it was back to the drawing board and some old-fashioned coding.

My next approach was to learn how to program the menu features by hand and to explore the uprated BROWSE command. I hoped these would eliminate much of the work, as handling menus of options and scrolling displays of data records had accounted for half the programming effort in the old system.

Programming the menus turned out to be quite a brave new departure from the *dBASE* style. Instead of the 'procedural' method of working, where you display the menu text, get the user's option and act on it, in *dBASE IV* you work in a way which could loosely be termed 'object oriented'.

Firstly, this involves describing a menu – its name, size, screen position plus a list of options and the actions to take for them. The menu then becomes an 'object' stored in memory, for you to activate when you want to.

Activation causes the menu to pop up and the appropriate programs to be run (or further menus activated) in response to the various options, returning back to the menu each time and continuing until



you 'deactivate' it. You can build 'trees' of menus which call other menus and programs, then kick the whole lot off with one ACTIVATE MENU statement.

The BROWSE command was also pretty good. I am interested in providing users with spreadsheet-like tables of database records to scroll through and choose from, and BROWSE was the original database table-builder. Nowadays it's more like a mini-program generator, with options for making data fields read-only, displaying calculated values (such as 'NET + VAT') and 'locking' the leftmost fields in spreadsheet fashion while the rest of the record scrolls horizontally against them.

Best of all, you can toggle between the BROWSE table and a full-screen display of the current record. Also, with some delving I found that the display could be my own customised one, complete with data validation routines.

While the new BROWSE is good, it fails miserably in one crucial respect. In many business applications you want to browse through only those records which belong to a particular group, such as invoice records having the same account number as the customer record you're showing at the top of the screen.

In *dBASE* you can only do this by using the SET FILTER command; and because of the way that works, a single press of the [PgUp] key can leave you sitting there for minutes on end while it searches through the whole database looking for more qualifying records – frustrating for the user and a disaster on a network as megabytes of useless data fly back and forth across the wires.

It's a problem made more frustrating by the existence of an obvious solution – a BROWSE WHILE command, which stopped looking as soon as it found a record with the 'wrong' key value. This would depend on the database being appropriately indexed – but then so does *dBASE's* SET RELATION TO command, which forms the cornerstone of its 'relational' data handling. Incredibly there is an EDIT WHILE command which does exactly this, but in the less useful context of a full-screen display. Action, please, A-T.

In the end I gave up and settled for SET FILTER (our files are small, so the delay is 'only' 30 seconds or so), and moved on to a much happier encounter with the *dBASE IV* report generator.

I'm not normally a fan of report generators, but this one is excellent. It's easy to use, handles end-of-group control breaks (account sub-totals and the like) well, and allowed me to do some cross-referencing between data files without resorting to the SET RELATION TO command, surely the most awkwardly designed programming instruction in the history of computing.

After about five days I had a system. First, we installed it on the network using *dBASE* in multi-user mode – one copy on the central server, and from there we loaded to each workstation. It was a performance disaster.

Admittedly we were only using a (temporary) 286 for the server (with *MainLAN GTi* as the network operating system), but it was bad in a way that not even an emergency 486 was going to cure. Deterred, I checked on the RUNTIME package which comes with the *Developers Edition.* It lets you run pre-written *dBASE* programs directly from the workstations (using the royalty-free runtime software), using the network only for access to the data files.

We tried it, and it worked. It eats up around 1.5 Mbytes of disk space on each workstation, but performance is about three times as fast (though still not great). The key point is that the data file sharing and record locking works in the same way as in a multi-user setup.

The moral – if you want to run pre-written programs on the workstations, use the runtime software, not multi-user *dBASE.* Even better, wait for Ashton-Tate to produce the stand-alone .EXE file compiler, that might bring performance improvements as well.

The current state of play? It's running all day, it doesn't crash, the record locking works magnificently, and the performance is so-so, deteriorating visibly as more records are added to the database. This is partly network-related, so we're upgrading to a 386 server and either *Netware 386* or *MainLAN 386,* depending on which one Steve Patient rates best. I'll report back later. ●